Proc Ling Soc Amer 2, 1:1-15. https://doi.org/10.3765/plsa.v2i0.4009.

# Learnability and falsifiability of Construction Grammars

Jonathan Dunn*

**Abstract**. The strength of Construction Grammar (CxG) is its descriptive power; its weakness is the learnability and falsifiability of its unconstrained representations. Learnability is the degree to which the optimum set of constructions can be consistently selected from the large set of potential constructions; falsifiability is the ability to make testable predictions about the constructions present in a dataset. This paper uses grammar induction to evaluate learnability and falsifiability: given a discovery-device CxG and a set of observed utterances, its learnability is its stability over sub-sets of data and its falsifiability is its ability to predict a CxG.

**Keywords**. construction grammar, grammar induction, discovery-device grammar

**1. CxG As Discovery Device.** Construction Grammars (CxG; Goldberg, 2006; Langacker, 2008) are descriptive grammars containing sets of symbolic form-meaning mappings. The strength of CxG is its descriptive power: for example, it combines item-specific representations of idioms and usage-constraints with fully-schematic representations by incorporating multiple levels of abstraction. Its weakness, however, is that the learnability and falsifiability of these unconstrained representations is difficult to evaluate. Learnability here is the degree to which the optimum set of constructions can be consistently selected from the large set of potential constructions; falsifiability is the ability to make testable predictions about the constructions present in a dataset. Traditional introspective approaches to CxG have no procedure for (i) selecting one potential constructional representation over another and (ii) making testable predictions about the constructions present in the grammar of a particular language. This paper uses an unsupervised construction grammar induction algorithm (e.g., Forsberg, et al., 2014; Dunn, 2016) to formalize CxG sufficiently to allow learnability and falsifiability to be evaluated at the level of learning. Thus, given a discovery-device CxG (c.f., Chomsky, 1957) and a set of observed utterances, its learnability is its stability over sub-sets of data and its falsifiability is its ability to predict a grammar of constructions.

The discovery-device grammar (C2xG) produces constructions as sequences of slots that (i) may be filled by complex nested constituents that take on the properties of their head and (ii) may be constrained using lexical, syntactic, and semantic information (C2xG has an unsupervised shallow constituency parser and a distributional semantic dictionary as sub-components). Given an observed utterance, the hypothesis space of potential constructions contains all observed sequences of simple or complex slots defined by all types of representation (lexical, syntactic, semantic). This is a very large hypothesis space and illustrates the underlying problem of learnability. These potential constructions are represented using a vector of multi-unit association values based on the $\Delta P$ (Gries, 2013), the direction-specific adjusted probability of co-occurrence: the left-to-right $\Delta P(X|Y) = p(X_P|Y_P) - p(X_P|Y_A)$, where $X_P$ indicates $X$ is present and $X_A$ that $X$ is absent. The learnability challenge is to consistently select the same optimum

* Data and code supporting this paper are available at www.jdunn.name. This research was supported in part by an appointment to the Visiting Scientist Fellowship at the National Geospatial-Intelligence Agency (NGA) administered by the Oak Ridge Institute for Science and Education through an interagency agreement between the U.S. Department of Energy and NGA. The views expressed in this publication are the author's and do not imply endorsement by the Department of Defense or the National Geospatial-Intelligence Agency. Author: Jonathan Dunn, Illinois Institute of Technology (jdunn8@iit.edu).

grammar out of the very large hypothesis space; here this is done using a metric that balances coverage and model complexity: $-log(C^2/S)$, where $C$ is the coverage of the hypothesized grammar on a test set and $S$ is its size in number of constructions (c.f., Goldsmith, 2006); this algorithm operates on a large number of potential grammars generated by applying heuristic classifiers to the association vectors.

Learning and evaluation are performed on the ukWac corpus (Baroni, et al., 2009), with sub-sets of 1 million sentences used to evaluate learnability. The hypothesis space across three sub-sets has only 40.56% agreement; the learning-based grammar averages 100% agreement while a rule-based baseline averages only 48.71% agreement, little more than the agreement in observed representations. This shows that a variant of CxG formalized as a discovery-device grammar does, in fact, produce learnable representations. Falsifiability follows from this because the learned grammar makes a testable hypothesis about actual vs. non-actual constructions. These results are discussed further in Section 2.5. The second section of this paper presents and evaluates the construction learning algorithm and the third section presents and evaluates the constituency parser that allows complex constituents to fill individual slots within a construction.

**2. Construction Grammar Induction.** Constructions are productive and schematic form-meaning mappings, the inventory of which represents the grammatical structure of a language (e.g., Goldberg, 2006; Langacker, 2008). This section presents an unsupervised algorithm for learning the optimum set of constructions to describe a corpus of observed language. The unsupervised nature of this work distinguishes it from previous computational approaches that rely on hand-crafted representations of constructions (Zadrozny, et al., 1994; Bryant, 2004; Micelli, 2006; Chang, et al., 2012; Steels, 2004, 2012) that (i) do not scale across dialects and languages and (ii) can tell us nothing about the learnability of construction grammars. This work improves upon previous unsupervised approaches (Wibble & Tsao, 2010; Forsburg, et al., 2014) in four ways: (1) it allows any type of complex constituent to fill a construction slot as a single entity; (2) it uses distributional semantics to support semantic selectional restrictions for slots; (3) it uses a larger number of features representing potential constructions to aid in the selection of the optimum grammar; (4) it develops a learning algorithm for selecting the optimum grammar out of the very large hypothesis space of potential grammars using a metric that balances both model coverage and model complexity. The evaluation shows that each of these improvements has a significant impact on the learned grammars in terms of both explanatory power and stability across different sub-sets of data.

Every observed linguistic expression has many potential constructional representations, a side-effect of the relatively unconstrained representations produced by CxG. For example, the utterance in (1a) could be described using a purely unit-based syntactic representation in (1b), a constituent-level syntactic representation in (1c), a semantically-constrained representation in (1d), and a lexically-fixed representation of the non-compositional material in (1e). In CxG, these and many other competing representations are potential constructions. The challenge is to select the smallest and most generalizable set of representations out of this large hypothesis space of potential representations.

(1) a. “Bill gave his neighbor a piece of his mind.”
b. [NN – VB – PRN – NN – DT – NN – PREP – PRN – NN]
c. [NP – VB – NP – NP]
d. [NP <ANIMATE> – VB <TRANSFER> – NP <ANIMATE> – NP]
e. [“a piece of his mind”]

The essential problem, then, is to distinguish between (i) potential but unproductive representations and (ii) actual representations present in the grammar that produced the observed language (c.f., Zuidema, 2006). The basic idea of CxG is that grammar is more than just a formal system of stable but arbitrary rules for defining well-formed sequences. Rather, grammar consists of meaningful constructions in the same way that a lexicon consists of meaningful words. This brings together two important premises: First, that grammar consists of meaningful symbolic units (e.g., Langacker's Cognitive Grammar); Second, that co-occurrence and distribution are indicators of meaning (e.g., Firth, 1957; Cilibrasi & Vitanyi, 2007). These premises suggest that constructions, like words, can be studied and defined as a set of co-occurring elements in a corpus. Co-occurrence is a measure of the relative productivity of competing representations; for example, we expect the more generalized constituent representation in (1c) to co-occur more significantly than the single unit representation in (1b) because there are many possible configurations like (1b) that are covered by the representation in (1c). We expect grammatical constructions to display internal co-occurrence that distinguishes them from unproductive representations.

2.1. REPRESENTING CONSTRUCTIONS**.** Constructions are schematic representations of potentially complex form-meaning mappings. This is best shown by example: the ditransitive in (2a) consists of an NN-PHRASE subject, a verb that indicates transfer of some sort, an indirect object NN-PHRASE that is confined to animate beings, and a direct object NN-PHRASE that is being transferred. Thus, this construction is defined in terms of both purely syntactic information (e.g., NN-PHRASE) and semantic selectional constraints (e.g., VB-PHRASE <TRANSFER>). Note that individual syntactic units are indicated by small caps (e.g., NN) and constituents by phrase types in small caps (e.g., NN-PHRASE). Semantic selectional restrictions are represented using domains enclosed in brackets (e.g., <ANIMATE>); these do not need to be attached to syntactic units. Lexical items are represented using quotation marks. The constructional representations are sequences of slots, each constrained by syntactic, semantic, and lexical restrictions at the word- or constituent-level. For example, the slot VB-PHRASE <TRANSFER> can be filled or satisfied by any verb constituent from that particular semantic domain (e.g., "give," "send," "sell"); this means that the observed linguistic expression in (2b) satisfies the slot requirements of the construction in (2a) and counts as an instance of that construction.

(2) a. [NN-PHRASE – VB-PHRASE <TRANSFER> – NN-PHRASE <ANIMATE> – NN-PHRASE]
b. "The child gave his brother a new book."
c. "give" – NN-PHRASE <ANIMATE> – "a break"
d. "Please give me a dollar."
e. "Please give me a break."

Constructions are posited at multiple levels of abstraction, so that more schematized representations like (2a) co-exist with item-specific representations like (2c). In this case, (2c) is a partially-fixed instance of the ditransitive that is not fully compositional. Thus, the linguistic expressions in (2d) and (2e) are produced by separate but related constructions that differ in their level of abstraction. An important hypothesis of CxG is that constructions range from fully-schematic to fully item-specific and that speakers are able to store both types together. Thus, part of the problem is finding the level of representation and abstraction which is most productive.

2.2. FINDING POTENTIAL CONSTRUCTIONS. The first step is to find all potential constructions in the dataset. As shown in Table 1, the basic idea is to iterate over subsets of the data and observe all sequences within *CS*(*max*), the parameter which limits the upper-bound of construction length

in number of slots. The algorithm then counts all possible representations of units (lexical, syntactic, semantic) and constituents (syntactic, semantic) which could potentially describe the sequences observed. This creates a very large number of potential representations and a frequency threshold is used to reduce this hypothesis space further because we expect actual constructions to be quite frequent. This first stage thus depends on three parameters: First, the size of the subsets of data; here it is 5k sentences per subset. Second, the maximum construction length influences the search space, which increases exponentially with each additional slot; here it is limited to 5 slots per construction. Third, the frequency threshold per subset is used to avoid storing and updating massive numbers of candidates; here it is 10.

| | |
|---|---|
| Def | *CS* = Contiguous sequence within single sentence of length between 2 and *CS*(*max*) |
| Def | *R*(*U*) = Representation by unit, where *R* = Lex, Syn, Sem |
| Def | *R*(*C*) = Representation by constituent, where *R* = Syn, Sem |
| Def | *PR* = Potential Representations, the set of *R*(*U*) and *R*(*C*) for *CS* in the current dataset |
| 1 | For each candidate in *PR*(*N*): Count candidate occurrences |
| 2 | If Candidate(Freq) > Frequency Threshold: Keep |
| 3 | For Kept Candidates: Calculate Association Vectors |

Table 1: Finding Potential Constructions

The algorithm models each linguistic expression with three types of representation (lexical, syntactic, semantic) at two levels of abstraction (word-level, constituent-level). First, the lexical level consists of word-forms as indicated in the orthography by whitespace. The syntactic representation of word-forms uses a part-of-speech tagger: here, RDRPosTagger (Nguyen, et al., 2016), the only supervised component of the algorithm. The word-level semantic representation is learned from the input corpus using word2vec (with 300 dimensions using cbow features; GenSim, Rehurek & Sojka, 2010) together with K-Means clustering ($k$ = 100) to produce a dictionary of domains for each word. For the constituent representation, an unsupervised constituent-parsing algorithm is used to identify (i) constituents and (ii) the head units of constituents, discussed further in Section 3. Identified constituents are treated as a single unit with the head unit supplying the syntactic and semantic information: the constituent "tall brown-eyed man" would be represented as a single unit, syntactically as an NN-PHRASE and semantically as a member of the ANIMATE domain (actual distributional domains are not named). The purpose of this is to allow complex constituents to fill a single slot as the output constructions are agnostic as to whether slots are filled by words or constituents.

2.3. MULTI-UNIT ASSOCIATION MEASURES FOR EVALUATING POTENTIAL CONSTRUCTIONS. At this point the algorithm has access to all potential representations together with their frequencies, the search space for actual constructions. Given only a single a sentence it is difficult to determine which representations are constructions and which are chance or non-optimum sequences. Given a large set of observations, however, we expect that actual constructions will rise to the top in terms of frequency and association. This is because they are both productive (thus more frequent) and meaningful (thus more associated). In other words, we expect that actual constructions will have a higher internal association than potential constructions because they represent meaningful grammatical units rather than chance co-occurrences. The algorithm creates a vector for each potential construction that contains several measures of association.

The basic measure of association is the $\Delta P$ (Gries, 2013; c.f., Gries, 2008, 2012), a direction-specific measure that can be calculated both left-to-right and right-to-left. It is based on the

probability that two units co-occur adjusted by the probability that one occurs without the other, with the ordering of the units determining the direction of association calculated. The pairwise measure is calculated as follows: Let $X$ be a unit of any representation and $Y$ be any other unit of any representation, so that $X_A$ indicates that unit $X$ is absent and $X_P$ indicates that unit $X$ is present. The LR measure is $p(X_P|Y_P) - p(X_P|Y_A)$ and RL is $p(Y_P|X_P) - p(Y_P|X_A)$. While $\Delta P$ usually measures the relationship between two words, it is converted into a multi-unit measure in four ways, as discussed further elsewhere (Dunn, 2016). Taken together these measures are used to represent how strongly a particular potential grammatical representation is co-located. More cognitively entrenched constructions are expected to be more associated, providing a measure for which potential representations belong in the final grammar.

2.4. DISTINGUISHING POTENTIAL AND ACTUAL CONSTRUCTIONS. At this point the algorithm has (i) the set of all possible constructional representations that pass a frequency threshold and (ii) a vector of features representing each. We expect actual constructions to be outliers in the sense that they represent meaningful or significant co-occurrences while other candidates are either chance sequences or contain non-optimum representation types. This section presents two algorithms for distinguishing potential and actual constructions: First, a series of heuristic rules with fixed thresholds; Second, a learning algorithm that seeks to maximize the coverage of the grammar while minimizing its size. The second algorithm uses the first to generate a large number of candidate grammars which are evaluated against one another on unlabeled data.

The first stage of the algorithm uses a series of heuristic rules with fixed thresholds to prune non-constructions from the grammar. These heuristics are based on thresholds applied to the vectors of association values: only potential constructions which exceed a given threshold for a given measure are allowed into the grammar. In addition, horizontal pruning finds potential constructions contained in other candidates and keeps only the longest possible candidate (c.f., Wible & Tsao, 2010). Thresholds for the heuristic classifier are set relative to the mean and standard deviation of features before pruning. There are a large number of potential grammars generated by these heuristics. We take these as hypotheses: each association threshold posits that the set of constructions it produces best describes the dataset.

The learning algorithm searches across these potential grammars to find the optimum grammar (i.e., the optimum set of constructions). Because there are 13 independently thresholded heuristics (left-to-right and right-to-left measures are thresholded together), each with a fixed range of possible thresholds, the algorithm produces a total hypothesis space of 1.5 million potential grammars. The problem is to search across this space and measure the quality of each potential grammar. The ultimate goal is to find a grammar that explains all observed linguistic expressions. On the one hand, the full set of potential constructions would explain all observations but make few generalizations. The optimum grammar, on the other hand, balances the need to explain all observations with the need to posit as few constructions as possible. The grammar learning algorithm, shown in Table 2, balances grammar size with coverage by minimizing the objective function $-log(C^2/S)$, where $C$ is the grammar's coverage and $S$ is its size. For example, this measure gives a grammar with 85% coverage containing 500 constructions a value of 2.84; 85% coverage and 800 constructions 3.04; 95% coverage and 1,100 constructions 3.08. Coverage is calculated independently for each construction on a word-by-word basis: each word described by a construction contributes to its coverage measure. A grammar's coverage is the total coverage of its parts.

| | |
|---|---|
| Def | $P$ = Set of candidates or potential constructions |
| Def | $Va(P)$ = Vector of association values representing $P$ |
| Def | $h$ = Parameter settings for each association measure |
| Def | $Gh(Va(P))$ = Potential grammar with parameters $h$ for association vector $Va$ for candidates $P$ |
| Def | $C$ = Words in training corpus explained by $Gh(Va(P))$ / All words in training corpus |
| Def | $S$ = Size of $Gh(Va(P))$ in number of $P$ |
| 1 | For all possible $Gh$ given $Va(P)$: The optimum grammar minimizes $-log(C^2/S)$ |

Table 2: Evaluating Potential Grammars

2.5. EVALUATING LEARNED GRAMMARS. This section evaluates the grammar learning algorithm using data from the ukWac corpus of web-crawled British English (Baroni, et al., 2009), testing how well construction grammars can be learned from noisy and informal data. The distributional dictionary was learned using the entire dataset (2 billion words). Separate subsets were used for learning the constituent-parser (800k sentences), for identifying potential constructions (1 million), for learning the optimum grammar (100k), and for the evaluation (100k).

| Method | Total | 2 Slots | 3 Slots | 4 Slots | 5 Slots | Coverage | $C^2/S$ | $C^2/S^2$ |
|---|---|---|---|---|---|---|---|---|
| PS-Only Baseline | 608 | 80 | 281 | 184 | 53 | 0.904 | 2.871 | 5.655 |
| Rule-based, Low | 4,472 | 2,397 | 1,627 | 357 | 81 | 2.777 | 2.763 | 6.413 |
| Rule-based, High | 821 | 277 | 297 | 184 | 53 | 1.114 | 2.820 | 5.735 |
| Learning-based, Low | 17,048 | 792 | 2,775 | 4,536 | 8,935 | 45.024 | 0.924 | 5.156 |
| Learning-based, High | 7,645 | 519 | 1,518 | 2,220 | 3,378 | 33.010 | 0.846 | 4.729 |
| No Constituents | 7,645 | 519 | 1,518 | 2,220 | 3,378 | 14.937 | 1.534 | 5.001 |

Table 3: Evaluation of Grammar Learning Methods

Several variants are compared; first, the unsupervised constituency-parser provided to the algorithm can be viewed as a set of syntactically-defined constructions and this provides the first baseline. The next baseline is the heuristic rules for generating possible grammars with fixed thresholds. Two variants of threshold ranges are used: Low ($\mu$,$\mu+\sigma$, Off) and High ($\mu+\sigma$,$\mu+2\sigma$, Off). These are compared with the learning algorithm which searches over the same two sets of possible thresholds for the optimum grammar. In addition to the metric described above $(C^2/S)$, a variant is added to further penalize large grammars: $C^2/S^2$. Taken together, these baselines describe the performance of this algorithm against previous unsupervised approaches.

First, the rule-based models have mostly short constructions (maxing at length 2) while the learning-based models have mostly longer constructions (maxing at length 5). Second, the learning-based models have more constructions than their rule-based counter-parts, but also have disproportionately higher coverage. Thus, the optimum grammar is the learning-based model searching over high thresholds, with a final score of 0.924, while the best rule-based model has a score of 2.763 (lower is better). Even using the additional metric, which adds a further penalty to large grammars, the same method performs best. This shows large improvements over the baseline and represents a significant advance over previous attempts to select the best construction representations (Forsberg, et al., 2014; Wible & Taso, 2010) which relied entirely on rule-based methods. Further, previous approaches allow only limited generalization across complex constituents; to evaluate the importance of this addition, the final variant in Table 3 shows the performance without allowing constituents to fill slots. These two improvements made by the algorithm show large improvements over previous approaches.

| Size: | 2 | | | 3 | | | 4 | | | 5 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | *In* | *All* | *Mixed* | *In* | *All* | *Mixed* | *In* | *All* | *Mixed* | *In* | *All* | *Mixed* |
| Syntactic Slots | 282 | 36 | 246 | 1,039 | 157 | 882 | 1,910 | 94 | 1,816 | 3,152 | 34 | 3,118 |
| Semantic Slots | 295 | 54 | 241 | 979 | 93 | 886 | 1,854 | 46 | 1,808 | 3,252 | 90 | 3,162 |
| Lexical Slots | 213 | 12 | 201 | 446 | 0 | 446 | 655 | 0 | 655 | 879 | 0 | 879 |

Table 4: Break-Down of Optimum Grammar

What sorts of constructions does this optimum grammar contain and how much do we gain by incorporating semantic representations as slot fillers? A descriptive break-down is given in Table 4 for each length (In: constructions containing a certain type; All: constructions containing only a certain type; Mixed: constructions containing a certain type among others). First, the three representation types are equally mixed in constructions of length 2, but the relative presence of lexical items declines quickly as construction length increases. The mix of syntactic and semantic representations remains relatively stable. Second, most constructions have mixed representation types, with only 8% containing a single type. This shows that the use of multiple types of representation (e.g., semantic) improves grammar quality by adding possible dimensions of generalization (c.f., Kay & Fillmore, 1997; Goldberg, et al., 2004).

| Subsets: | 1 + 2 | 1 + 3 | 2 + 1 | 2 + 3 | 3 + 1 | 3 + 2 |
|---|---|---|---|---|---|---|
| Raw Candidate Agreement | 43.9% | 44.5% | 35.4% | 40.9% | 36.8% | 41.9% |
| Baseline Grammar Agreement | 32.3% | 32.5% | 71.4% | 74.7% | 77.3% | 74.1% |
| Optimum Grammar Agreement | 100% | 100% | 100% | 100% | 100% | 100% |

Table 5: Stability of Learned Grammars

Given the best grammar learning method, the next question is whether this method produces stable results across mutually-exclusive datasets. The intuitive observation is that children learn relatively similar grammars even though they observe different linguistic expressions while learning. Thus, an important criteria for the algorithm is that it produce relatively stable grammars. This experiment recreates the optimum grammar on two new sub-sets of the ukWac corpus and measures agreement between the subsets in terms of both potential constructions and final grammars. The idea is that the first type of agreement measures similarity between the raw observations and the second measures similarity between the learned representations. Agreement is calculated in a pairwise fashion, with one grammar acting as a gold-standard, as shown in Table 5. There is low agreement in the total hypothesis space across sub-sets, in the low 40s on average. However, there is perfect agreement between grammars found using the learning algorithm. The comparison shows that other methods (a rule-based baseline) are not able to consistently select the same constructions from the noisy hypothesis space.

**3. Unsupervised Catena-Based Constituent Parsing.** This section presents an unsupervised algorithm for building representations of base constituents from observed pairs of units. The key insight is to view constituents as sequences of catena-pairs, where a catena is a sub-constituent syntactic unit whose members share the same dominance relations but do not necessarily form a constituent themselves. Every constituent consists of one or more pairs of catena units. The algorithm thus has two basic tasks: First, determining for all pairs of units whether they are catena or non-catena-pairs; Second, determining for all catena-pairs how they can be joined to form constituents. The evaluation shows that the algorithm performs as well as a supervised alternative on the task of shallow constituent parsing. Because CB-CP works by joining together catena-pairs into their maximal projections (i.e., constituents) it does not require complete

structures for either learning or annotation. In other words, the algorithm is ideal for working with noisy and incomplete texts; here the learning and evaluation is performed using the web-crawled ukWac corpus of English (Baroni, et al., 2009).

The goal of the algorithm is to (1) identify basic constituents and (2) identify which basic constituents are interchangeable while (3) using only pairwise information. This building of constituents is similar to building CCG representations (Bisk & Hockenmaier, 2013) but makes reference only to the concepts of catena and catena-heads. At the same time, the goal is to produce typed and interchangeable constituents (unlike, for example, Seginer, 2007). Some comparable unsupervised approaches rely on additional information, such as prosody (Pate & Goldwater, 2011) or punctuation (Seginer, et al., 2007) that is not assumed to be available while others make use of non-pairwise sequences (e.g., Dennis, 2005; Ponvert, et al., 2010, 2011). The current output is a bracketing of shallow or base constituents, each a phrase of a certain type. Thus, "the dark clouds" in (3a) is a determiner phrase and behaves as a determiner phrase regardless of its internal structure. The problem is shown in (3b), where arbitrary sequences have been bracketed: some are incomplete constituents (e.g., "the dark" without "clouds") and others cross constituent boundaries (e.g., "filled the").

(3) a. [DT The dark clouds] [V filled] [DT the sky].
b. [The dark] [clouds] [filled the] [sky].

This illustrates the difficulty of building constituents in a pairwise fashion. For any given pair of units, there are three possibilities: (1) the pair can form a complete constituent; (2) the pair can cross constituent boundaries; (3) the pair can form a partial constituent that requires additional material. Although a pairwise approach reduces the complexity of the problem, most pairs cannot be identified directly as constituents. CB-CP overcomes this difficulty using the catena, a sub-constituent unit. A catena is "a word or combination of words the projections of which are continuous with respect to dominance" (Osborne, et al., 2012: 355). This means that only some catena are constituents but all constituents are catena. If two units form a catena-pair, they have the potential to be a constituent given additional units.

| | | | | |
|---|---|---|---|---|
| (4) | *String:* | a. "inside the blue house" | | |
| | *Constituents:* | b. [PP inside the blue house] | c. [DT the blue house] | |
| | *Catena-Pairs:* | d. (inside the) | e. (the blue) | f. (blue house) |

A constituent is "a catena that consists of a word plus all the words that that word dominates" (Osborne, et al., 2012: 359). Given the string in (4a), there are two constituents: a prepositional phrase in (4b) that contains a determiner phrase in (4c). There are three catena-pairs shown in (4d) through (4f). The essential idea of the algorithm is to use catena-pairs to reduce the complexity of the search for constituents. It examines all pairs of syntactic units or parts-of-speech ($P_1$, $P_2$) and determines which are catena-pairs. For example, in "The dog looked at the blue sky," the pairs "the dog", "the blue", and "blue sky" are all catena-pairs because they do not have a structural break between the two units. Of these, only "the dog" is a constituent on its own; "dog looked" is not a catena-pair and cannot be a constituent.

3.1. CATENA CONSTRAINTS. The primary assumption of the algorithm, A in Table 6, is that all heads project in a single direction. The term "head" refers to heads of catena-pairs; thus, an adjective can be the head of a catena-pair even if it never heads a constituent. The idea is that each pair ($P_1$, $P_2$) gets its identity from its head. Catena-pairs are partial projections and constituents are maximal projections of a head. Because all sequences in consideration are pairs

and language is one-dimensional, all units occupy either the left or the right end-point. The assumption, then, is that a given head will always occupy the same end-point in all its projections, whether partial or maximal: ($P_1$, X) or (X, $P_1$). In this notation, *X* refers to any unit which can occupy the left position given $P_1$: (X, $P_1$). The effect of this assumption is that all catena-pairs of a given type contain their head in the same end-point. A counter-example to this generalization could be adjuncts, as with the adverbs in (5a) and (5b) which occur both to the left and to the right of the verb. In the right position, however, they can be displaced as shown in (5c). Thus, it is plausible to analyze the right position as a separate particle and not part of a catena-pair with the verb, so that the V-ADV relation does not violate this assumption.

(5) a. “He slowly ran through the yard."
b. “He ran slowly through the yard."
c. “He ran through the yard slowly."

| |
|---|
| A: Heads Have One Direction<br>For all catena-pairs ($P_1$, $P_2$) in which $P_1 \neq P_2$:<br>Heads project from only one end-point<br>Left-Head is written as ($P_1$, $R_X$) and Right-Head written as ($L_X$, $P_1$) |
| B: Reverse Pairs Cannot Be Catena<br>For ($P_1$, $P_2$) where $P_1 \neq P_2$:<br>If ($P_1$, $P_2$) is catena, ($P_2$, $P_1$) is not catena<br>Either $P_1$ or $P_2$ would be head in both pairs: e.g., $P_1$ occupies both end-points as head |
| C: Catena-Pairs Must Have Head<br>For ($P_1$, $P_2$), if $P_1$ is R-Head and $P_2$ is L-Head:<br>($P_1$, $P_2$) cannot be a catena because neither unit projects the pair |

*Table 6: Constraints*

The second assumption, B in Table 6, is that the converse of a catena-pair cannot also be a catena-pair. This follows from the first assumption: if ($P_1$, $P_2$) is a catena-pair, either $P_1$ is a head that occupies the left end-point or $P_2$ is a head that occupies the right end-point. The reverse then cannot be a catena-pair because the head would project from multiple end-points. This assumption is used to remove pairs from consideration as catena: once we know that $P_1$ is either a left-head or a right-head, all candidate pairs requiring the opposite can be discarded. The third assumption, C in Table 6, is that a pair in which neither end-point is a head unit occupying its head position cannot be a catena-pair.

We evaluate these assumptions empirically by annotating all pairs of tags from the Penn Treebank (Santorini, 1990) as catena-pairs with single-direction heads or non-catena-pairs. The purpose is to determine (i) if such a gold-standard annotation is possible and (ii) whether it obeys the above assumptions. This analysis also provides a test set for distinguishing catena and non-catena-pairs. The gold-standard pairs show that a pairwise analysis that does not violate the assumptions is possible for English. The tag set has 33 unit types, for a total of 1,089 pairs. Only 870 of these pairs are observed; unobserved pairs are considered non-catena-pairs. Catena-pairs account for 268 (25%) of the total pairs; the majority (233 or 87%) are left-headed.

3.2. DETERMINING WHEN SUFFICIENT DATA HAS BEEN OBSERVED. The first component, shown in Table 7, learns and stores a matrix of association and frequency values for each pair of units; this matrix is used in the later stages as part of the classifier for identifying catena-pairs. The dataset is divided into chunks of *N* size (currently *N* = 5,000 sentences). For each chunk, association values are calculated with the chunk included and then the cosine distance between

the current and the previous association values is computed. This is a simple measure of the amount of change that the current chunk of data creates in the observed association between units. The algorithm's only required parameter is the stability threshold used to determine when to stop looking at more data for learning association between units. The amount of data required to reach different thresholds for stability is also shown in Table 7, along with the amount of time in seconds required to calculate the association matrix (using an Intel Core i7 860 @ 2.80 GHz). The purpose is to provide a principled stopping point. The ukWac corpus of web-crawled English is used as the dataset together with RDRPosTagger (Nguyen, et al., 2016); the time for all stages of the algorithm ($\Theta$=0.00001) is 46 min.

| Learning Pairwise Representations | | Time and Data Required | | |
|---|---|---|---|---|
| Def | *N* = Amount of data per learning iteration | $\Theta$ | Sentences | Seconds |
| Def | $\Theta$ = Association stability threshold | 0.001 | 60k | 133 |
| 1 | For *N* new data: | 0.0001 | 130k | 287 |
| 2 | For all pairs, calculate $\Delta P(LR)$ and $\Delta P(RL)$ | 0.00001 | 235k | 897 |
| 3 | C = cosine distance between new and old $\Delta P$ | 0.000001 | 840k | 1,894 |
| 4 | If $C<\Theta$: Return pairwise frequency matrix | | | |

Table 7: Algorithm for Learning Frequency Measures and Data / Time Required

3.3. IDENTIFYING CATENA-PAIRS. The second stage of the algorithm is to distinguish between catena and non-catena pairs. As shown in Table 9, this algorithm has two main steps: First, sort pairs by frequency, on the assumption that more frequent pairs are more likely to be members of the same catena; Second, iterate over the pairs applying a classification formula to identify a pair as a catena-pair; the threshold for identification lowers with each iteration. The reasoning behind this approach is that each identified catena-pair provides information about other potential catena-pairs and their head units. Thus, each identification overrules other potential identifications (c.f., the multi-pass sieve in Raghunathan, et al., 2010). The goal, then, is to give more likely catena-pairs precedence. The threshold for pair classification begins at its maximum potential point, the highest frequency for an individual pair, and is reduced after each iteration by 10% of the standard deviation in frequency values; the point is to spread out the evaluation of pairs using the frequency distribution. The algorithm stops when all pairs have been classified or the threshold reaches 0; pairs not identified as catena-pairs are labelled as non-catena-pairs.

The basic question for the catena-pair classifier is whether a given pair of units is or is not part of the same catena. Given a pair ($P_1$, $P_2$), the set of all pairs in which $P_1$ occupies the left end-point is represented as ($P_1$, X) and the set of all pairs in which $P_1$ occupies the right end-point is represented as (X, $P_1$). Given that in the current pair $P_1$ occupies the left end-point, if this is a catena-pair this sequence will predominate. Thus, $\Sigma[(P_1, X) – (X, P_1)]$ represents how much more frequent the expected sequence is than its reverse across all observed pairs, with ($P_1$, X) quantified by frequency. The same measure is calculated on $P_2$ as $\Sigma[(X, P_2) – (P_2, X)]$. These measures are combined as the pair status classifier.

The basic question for the catena-head classifier is whether $P_1$ or $P_2$ is the head of a given catena ($P_1$, $P_2$). Given Assumption A we proceed with the knowledge that the head always projects in the same direction and thus must always occupy the same end-point when acting as head. This allows us to leverage information about all other pairs to determine the status of the current head. Frequency is a less useful measure here because the head and non-head occur together in the sequence. Let $P_D$ be the unit occupying the end-point *D* in the current pair, where *D* is either left or right. Additionally, let $A(P_D)$ be the set of association values for $P_D$ in all pairs where *P*

occupies end-point *D*, with association values proceeding from *D*. The head direction classifier is then $\Sigma(\Delta P(LR)) - \Sigma(\Delta P(RL))$. Positive values indicate that the pair has a left-head and negative values indicate that the pair has a right-head.

| | |
|---|---|
| Def | $\Theta$ = Max Pair Frequency; $\delta = \sigma$(Pair Frequency)/10 |
| Def | $C_C = \Sigma[((P_1, X) - (X, P_1)) + ((X, P_2) - (P_2, X))]$ |
| Def | $H_C = \Sigma(\Delta P(LR)) - \Sigma(\Delta P(RL))$ |
| 1 | While $\Theta > 0$, For all pairs sorted by frequency: |
| 3 | If $P_1 = P_2$, Special Case: |
| 4 | Either [P + P = P] or [P \| P] |
| 5 | If $P_1 \neq P_2$: |
| 6 | If Known: Skip |
| 7 | If Unknown and $C_C < \Theta$: Check Head Classifier ($H_C$) |
| 8 | If Obeys Assumptions A, B, and C: |
| 9 | Save Catena Status, Head Direction, and Reverse Pair Status |
| 10 | Adjust classifier threshold: $\Theta = \Theta - \delta$ |

Table 8: Algorithm for Identifying Catena-pairs

When $P_1 = P_2$ there are two possibilities: First, that *P* combines with itself to produce *P*. This is the case with the adjectives in "the dark clouds" and "the large dark ominous clouds" where any number of adjectives can be combined to form an adjective phrase. Second, if *P* cannot combine with itself then (P, P) must be a non-catena-pair. For example, in "went in beyond the door" the prepositions "in" and "beyond" do not combine to produce a single preposition, as with the adjectives. There is a constituent boundary between them. Because units of a single type tend not to co-occur unless in the same syntactic unit, those pairs above the mean frequency for duplicate pairs are taken as permitting duplication.

| Random | TP | FP | TN | FN | Acc. |
|---|---|---|---|---|---|
| -FT, -HF | 25.0% | 25.0% | 25.0% | 25.0% | 50.0% |
| +FT, +HF | 41.7% | 8.4% | 34.3% | 15.4% | 76.0% |
| +FT, -HF | 41.7% | 8.4% | 34.3% | 15.4% | 76.0% |

Table 9: Evaluation of Catena-Pair Identification

We evaluate the catena identification algorithm under four conditions: First, with and without a pair frequency threshold (FT); Second, with and without head-filling (HF): if the predicted head in a predicted catena-pair is incompatible with previous head direction predictions, the remaining end-point is made the head if possible. The results, shown in Table 9, are given by instance so that more frequent pair types have more weight. First, the overall accuracy is much higher than the baseline and the largest source of error is false negatives. These are mostly infrequent but syntactically-acceptable catena-pairs. However, given that the algorithm proceeds by joining all possible pairs of catena-pairs, false negative errors are preferable to false positive errors. The different conditions do not significantly change performance.

3.4. BUILDING CONSTITUENTS FROM CATENA-PAIRS. Catena-pairs reduce the search space for constituents in two ways: First, only catena-pairs can be part of a constituent; Second, only linked sequences of catena-pairs can be constituents. The algorithm which takes identified catena-pairs and joins them together to create constituents is shown in Table 9. Given that catena and constituents are partial and maximal projections from a head, a given head can only project

out from the non-head end-point of the current sequence and any units that belong to the same catena-pair are part of the same structure. Heads continue to project through sequences of catena-pairs until that projection becomes maximal (i.e., a constituent). For any catena-pair ($P_1$, $P_2$), the pair can only join at the non-head end-point with other pairs sharing that end-point. For example, ($P_1$, $P_2$) where $P_1$ is head can join with any catena-pair ($P_2$, X). In Table 10, heads are indicated as *H* and non-heads as *X*, so that catena-pairs can take the form of either (*H*, *X*) or (*X*, *H*).

| | |
|---|---|
| 1 | For all Catena-pairs ($P_1$, $P_2$): |
| 2 | *Y* = Non-Head Unit with *X* if *X* Dominates *Y* |
| 3 | Join with catena-pairs sharing *X* |
| 4 | (*H*, *X*) joins with (*X*, *Y*) and (*X*, *H*) joins with (*Y*, *X*) |
| 6 | Joined Sequences = Constituents iff: |
| 7 | End of sequence path is reached: |
| 8 | i. Non-Head end-point *Y* projects toward *H* |
| 9 | ii. No further joins possible |
| 10 | Constraint: No single unit may repeat |

Table 10: Algorithm for Building Constituents

Consider the catena-pairs in (6a) through (6c), with asterisks marking heads. All three pairs combine on the shared non-head end-point creating the constituent in (6e). This is a constituent because it contains all combinations allowed on its path. A different path combines (6b) directly with the noun, as in (6d), creating the constituent in (6f). Joining stops when: (i) the non-head end-point projects towards the head, preventing further joins (c.f., separators and sub-separators in Santamaria & Araujo, 2010; and boundaries more generally in Spitkovsky, et al., 2012); (ii) no further catena-pairs exist which share the appropriate end-point. Note that (6b) and (6d) do not combine but represent different paths from DT.

*Catena-Pairs:* (6a) (*PREP,DT) (6b) (*DT, ADJ) (6c) (ADJ, *N) (6d) (DT, *N)
*Constituents:* (6e) [PREP, DT, ADJ, N](6f) [PREP, DT, N] (6g) [PREP, N] (6h) [PREP, ADJ, N]

A further constraint prevents the algorithm from falling into loops: no units are allowed to repeat. This avoids endless loops where a sequence (*A*,*B*,*A*) continues to feed itself. The final stage of the joining algorithm is to prune the predicted phrase structure rules by returning to the original dataset and pruning unobserved rules. The iter-prune variant prunes unobserved candidate sequences at the end of each iteration and the end-prune variant prunes after the joining algorithm has completed.

3.3. EVALUATION. We take two approaches to evaluation: First, a constituent-by-constituent evaluation examines the accuracy of predicted constituents: these can be correct, too small, too large, or of the wrong type (i.e., the wrong head is identified). Second, a sentence-by-sentence evaluation looks at the constituent bracketing for an entire sentence, thus evaluating how many constituents fail to be identified; this evaluation includes the bracketing of nested constituents. Two baselines are used: First, the constituents produced using the gold-standard catena-pairs, representing the performance of the catena-joining algorithm with perfect input. Second, the chunking output of Apache OpenNLP, taken as the gold-standard supervised baseline.

The constituent-by-constituent evaluation in Table 11 looks at the accuracy of predicted constituents using 10k test sentences from the ukWac corpus. For each system 500 predicted constituents are chosen at random and evaluated. If the prediction is a correct or plausible analysis (in the case of ambiguity) it is marked as correct. Errors are classified into three

categories: Too Small (e.g., predicting "blue sky" instead of "the blue sky"), Too Large (e.g., predicting "the sky in" instead of just "the sky"), and Wrong Type (e.g., predicting "the blue sky" to be an adjective phrase). All constituents evaluated contain at least two units. Surprisingly, the gold-standard-pair variant of CB-CP has a slightly higher number of correct constituents than the fully-supervised baseline. The interesting difference is in the breakdown of errors: the unsupervised system makes a much larger number of type errors than the others because it must also predict which units are heads. For example, one of the unsupervised system's rules assigns DT-JJ pairs to the head JJ, a mistake. The iter-prune variant of the fully unsupervised CB-CP achieves the highest unsupervised accuracy for shallow constituent parsing; at 84.6% this is essentially the same performance as the supervised baseline, 84.0%. The gold-pair variant has the highest accuracy overall at 86.8%.

| | Constituent-By-Constituent | | | | Sentence-By-Sentence | |
|---|---|---|---|---|---|---|
| | *Correct* | *Too Small* | *Too Large* | *Wrong Type* | *Accuracy* | *Coverage* |
| CB-CP, Learned Pairs, End-Prune | 80.4% | 12.6% | 01.6% | 05.4% | 81% | 57% |
| CB-CP, Gold Pairs, End-Prune | 86.8% | 13.2% | 00.0% | 00.0% | 85% | 67% |
| CB-CP, Learned-Pairs, Iter-Prune | 84.6% | 06.8% | 00.0% | 08.6% | 85% | 57% |
| CB-CP, Gold-Pairs, Iter-Prune | 81.8% | 18.0% | 00.2% | 00.0% | 81% | 69% |
| OpenNLP Chunker | 84.0% | 08.2% | 06.8% | 01.4% | 85% | 58% |

Table 11: Evaluation Against Supervised Baseline

This first evaluation does not penalize systems for missing constituents; thus, it is possible that the CB-CP's comparison to the supervised baseline disguises a large number of false negatives. The second evaluation looks at predicted base-constituent bracketing for whole sentences using two measures: first, accuracy of predicted constituents and, second, coverage as percentage of the sentence within constituents of more than one unit. When compared for the same sentences across different systems, this provides a measure of how many constituents are missed. This measure of coverage is used to avoid comparing the algorithms to a single gold-standard corpus, allowing analyses that differ but remain acceptable. The results are calculated for 50 sentences from the test set and shown in Table 11. The unsupervised iter-prune variant achieves the same accuracy as both baselines at 85.0%. However, it has slightly lower coverage than the supervised baseline and significantly lower coverage than the semi-supervised CB-CP. Thus, for shallow constituent parsing the fully supervised CB-CP matches the performance of the supervised baseline and the semi-supervised CB-CP exceeds it.

| | Catena-Pair Identification | | | | | Constituent Identification | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Subset 1 | Subset 2 | Subset 3 | Subset 4 | | Subset 1 | Subset 2 | Subset 3 | Subset 4 |
| Subset 1 | – | 96.1% | 98.0% | 98.8% | Subset 1 | – | 61.1% | 62.9% | 66.2% |
| Subset 2 | 96.1% | – | 97.5% | 97.0% | Subset 2 | 72.5% | – | 69.2% | 69.1% |
| Subset 3 | 98.0% | 97.5% | – | 97.9% | Subset 3 | 70.5% | 65.3% | – | 64.8% |
| Subset 4 | 98.8% | 97.0% | 97.9% | – | Subset 4 | 71.1% | 62.5% | 62.1% | – |

Table 12: Stability Across Sub-Sets

The next question is whether CB-CP performs consistently when learned on mutually-exclusive datasets. The best-performing fully-unsupervised iter-prune variant is used to measure stability. Agreement is shown across four instances, each learned on a mutually-exclusive set of 400k sentences from the ukWac corpus. Agreement is calculated using pairwise accuracy: $A(1,2)$ = Items from Grammar 1 in Grammar 2 / Items in Grammar 1. This direction-specific measure is shown for all combinations of sub-sets in Table 12 for both catena identification and constituent

identification. We expect that catenae-pair identifications will show high agreement: the set of units that belong to the same syntactic unit should be largely stable across observed usage. This is the case, with agreement on average above 97% across subsets. Agreement for phrase structure rules (i.e., for schematic representations of possible constituents) is much lower. The rules produced by the grammar take the form of NNS =>DT–JJR–NNS, specifying both the head and the phrase structure. We do expect, then, a certain degree of variation in the set of phrase structure rules observed in different subsets of a corpus; this is what is shown in Table 12. Given the high agreement in catenae-pair identifications, the grammar would combine the pairs to produce the same set of phrase structure rules, except that unobserved rules are pruned.

**4. Summary**. This paper has presented discovery-device grammars for both CxG and for identifying constituents for purposes of filling slots within a CxG grammar. This type of discovery-device grammar provides a way to evaluate both the falsifiability and learnability of the representations posited by the CxG paradigm.

**References**


Baroni, M., Bernardini, S., Ferraresi, A., and Zanchetta, E. 2009. "The WaCky Wide Web: A Collection of Very Large Linguistically Processed Web-crawled Corpora." *Language Resources and Evaluation*, 43: 209-226. 26.

Bisk, Y. and Hockenmaier, J. 2013. "An HDP Model for Inducing Combinatory Categorial Grammars." *Transactions of the Association for Computational Linguistics*, 1: 75-88.

Bryant, J. 2004. "Scalable Construction-Based Parsing and Semantic Analysis." *Proceedings of the Second Workshop On Scalable Natural Language Understanding*. 33-40.

Chang, N.; De Beule, J.; and Micelli, V. 2012. "Computational Construction Grammar: Comparing ECG and FCG." In Steels, L. (ed.), *Computational Issues in Fluid Construction Grammar*. Springer: Berlin. 259-288.

Chomsky, N. 1957. *Syntactic Structures*. The Hague: Mouton & Co.

Cilibrasi, R. and Vitanyi, P. 2007. "The Google Similarity Distance." *IEEE Trans. on Knowl. and Data Eng.* 19(3): 370-383.

Dennis, S. 2005. "An Exemplar-based Approach to Unsupervised Parsing." *Proceedings of the 27th Conference of the Cognitive Science Society*. 583-588.

Dunn, J. 2016. "Computational Learning of Construction Grammars." *Language & Cognition.*

Firth, J. 1957. *Papers in Linguistics, 1934-1951*. Oxford: Oxford University Press.

Forsberg, M.; Johansson, R.; Backstom, L.; Borin, L.; Lyngfelt, B.; Olofsson, J.; and Prentice, J. 2014. "From Construction Candidates to Constructicon Entries: An Experiment using Semi-automatic Methods for Identifying Constructions in Corpora." *Constructions and Frames*, 6(1): 114-135.

Goldberg, A. 2006. *Constructions at Work: The Nature of Generalization in Language*. Oxford: Oxford University Press.

Goldberg, A.; Casenhiser, C.; and Sethuraman, N. 2004. "Learning Argument Structure Generalizations." *Cognitive Linguistics*, 15: 289-316.

Goldsmith, J. 2006. "An Algorithm for the Unsupervised Learning of Morphology." *Natural Language Engineering*, 12(4): 353-371.

Gries, S. 2008. "Dispersions and Adjusted Frequencies in Corpora." *International Journal of Corpus Linguistics*. 13(4): 403-437.

Gries, S. 2012. "Frequencies, Probabilities, and Association Measures in Usage- / Exemplar-based Linguistics: Some Necessary Clarifications." *Studies in Language*. 11(3): 477-510.

Gries, S. 2013. "50-something Years of Work on Collocations: What Is or Should Be Next." *International Journal of Corpus Linguistics*. 18(1): 137-165.
Kay, P. and Fillmore, C. 1999. "Grammatical constructions and linguistic generalizations: the What's X Doing Y? construction." *Language*, 75(1): 1-33.
Langacker, R. 2008. *Cognitive Grammar: A Basic Introduction*. Oxford: Oxford UP.
Micelli, V. 2006. "Searching for Grammar Right." *Proceedings of the 3rd Workshop on Scalable Natural Language Understanding*, Association for Computational Linguistics. 57-64.
Nguyen, Dat Quoca; Nguyen, Dai Quocb; Pham, Dang Ducc; and Pham, Son Baod. 2016. "A robust transformation-based learning approach using ripple down rules for part-of-speech tagging." *AI Communications* 29(3): 409-422.
Pate, J. and Goldwater, S. 2011. "Unsupervised Syntactic Chunking with Acoustic Cues: Computational Models for Prosodic Bootstrapping." *Proceedings of the 2nd Workshop on Cognitive Modeling and Computational Linguistics*. 20-29.
Ponvert, E.; Baldridge, J.; and Erk, K. 2010. "Simple Unsupervised Identification of Low-Level Constituents." *Proceedings of IEEE 4th Intl. Conference on Semantic Computing*. 24-31.
Ponvert, E.; Baldridge, J.; and Erk, K. 2011. "Simple Unsupervised Grammar Induction from Raw Text with Cascaded Finite State Models." *Proceedings of ACL*. 1,077-1,086.
Raghunathan, K.; Lee, H.; Rangarajan, S.; Chambers, N.; Surdeanu, M.; Jurafsky, D.; Manning, C. 2010. "A Multi-pass Sieve for Coreference Resolution." *Proceedings EMNLP*. 492-501.
Rehurek, R. and Sojka, P. 2010. "Software Framework for Topic Modelling with Large Corpora." *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. European Language Resources Association, Paris: 45-50.
Santamaria, J. and Araujo, L. 2010. "Identifying Patterns for Unsupervised Grammar Induction." *Proceedings of CoNLL 2010*. 38-45.
Santorini, B. 1990. "Part-of-Speech Tagging Guidelines for the Penn Treebank Project (3rd Revision)." *Technical Reports (CIS): Paper 570*. University of Pennsylvania.
Seginer, Y. 2007. "Fast Unsupervised Incremental Parsing." *Proceedings of ACL*. 384-391.
Spitkovsky, V.; Alshawi, H.; Jurafsky, D. 2012. "Three Dependency-and-Boundary Models for Grammar Induction." *Proceedings of EMNLP and CoNLL 2012*. 688-698.
Steels, L. 2004. "Constructivist Development of Grounded Construction Grammar." *Proceedings of ACL*. 9-16.
Steels, L. 2012. "Design Methods for Fluid Construction Grammar." In Steels, L. (ed), *Computational Issues in Fluid Construction Grammar*. Berlin: Springer. 3-36.
Vincze, V.; Zsibrita, J.; and Istvan, N. 2013. "Dependency Parsing for Identifying Hungarian Light-verb Constructions." *Proceedings of the Intl. Joint Conference on Natural Language Processing*. 207-215.
Wible, D. and Tsao, N. 2010. "StringNet as a Computational Resource for Discovering and Investigating Linguistic Constructions." *Proceedings of the Workshop on Extracting and Using Constructions in Computational Linguistics*. 25-31.
Zadrozny, W.; Szummer, M.; Jarecki, S.; Johnson, D.; & Morhenstern, L. 1994. "NL Understanding with a Grammar of Constructions." *Proceedings of COLING*. 1,289-293.
Zuidema, W. 2006. "What Are the Productive Units of Natural Language Grammar? A DOP Approach to the Automatic Identification of Constructions." *Proceedings of CoNLL*. 29-36.